Exploiting HHPC for Parallel Discrete Event Simulation

Nael Abu-Ghazaleh
Computer Science Department,
Binghamton University, Binghamton, NY
nael@cs.binghamton.edu

Richard Linderman, Robert Hillman, and James Hanna
Air Force Research Laboratory, Information Directorate, Rome, NY
{linderman, hillman, hanna}@rl.af.mil

Abstract

Parallel Discrete Event Simulation (PDES) is an important application in use in many DoD projects; for example, PDES is used in large-scale war-gaming, and in complex system design, analysis and verification. Improving PDES performance and capacity allows faster simulation times and more extensive analysis of more detailed models. These benefits are not application-specific: they should reflect to any application that uses the improved simulation kernel. In this work, we overview our efforts for optimizing PDES in a Heterogeneous High Performance Computing (HHPC) environment. We profile the SPEEDES simulator and identify several opportunities. We report on our experiences on two fronts: (1) optimizing the communication subsystem—a critical system for PDES since it is a fine-grained application and (2) exploring the use of augmented FPGA boards to accelerate simulation. While such approaches have been attempted for sequential and data path intensive applications, we believe that their use in clustered environments is novel. Both efforts are works in progress; we present our designs and some preliminary analysis results. For example, removing the centralized communication server from event message exchange path with a number of other small improvements to the simulation cycle, improved performance by an average of 20% performance improvement for one of our large benchmarks.

1. Introduction

Computer modeling and simulation have grown to become a powerful approach for complex system design and analysis. Parallel discrete event simulation (PDES) is an approach to parallelizing simulation to increase its performance and capacity, allowing the simulation of bigger more detailed models and more interesting scenarios in a given time budget. PDES underlies several areas of interest for the DoD including war-gaming, planning and decision making, and complex system design and analysis, including both hardware and software systems.

In PDES, a simulation model is partitioned across several logical simulator processes (or LPs). Each LP processes its events in time-stamped order. Synchronization among different LPs may be achieved using one of two major approaches: (1) Conservatively: an event at an LP is processed only if all other LPs guarantee that it can be processed safely (no events earlier than it will be generated to that LP); (2) Optimistically: LPs process events without concern for causality. Events received from other LPs, with a time stamp earlier than the current simulation time, signal a causality error. Such errors are recovered from by rolling back the local simulation state to a time earlier than the received straggler event messages that were sent out erroneously. To be able to achieve this synchronization, each LP must periodically checkpoint its state and event information. Checkpoints are garbage collected when they are no longer needed (when the global simulation time has passed them). This requires computing the Global Virtual Time (GVT) of the simulation to determine which history information may be garbage collected. Fujimoto[1] wrote an excellent survey on PDES and PDES optimization approaches.

We describe our efforts in exploiting a Heterogeneous High Performance Cluster (HHPC) to accelerate the performance of a PDES simulation engine. We start with the SPEEDES simulator: a state-of-the-art commercial parallel simulator in use in several DoD efforts. We first profile the SPEEDES using some large-scale applications developed within in-house projects. Based on this information, we identified some performance bottlenecks and opportunities for exploiting HHPC resources. We report our experiences in two primary directions:

1. Optimizing the communication subsystem: the standard implementation of SPEEDES uses TCP/IP and a centralized server for all event and group communication

messages. PDES is a fine-grained application that is limited by the communication performance both in exchanging events and in synchronization. Furthermore, the synchronization algorithm used by SPEEDES throttles the simulation relative to the global simulation time (or GVT, the Global Virtual Time) to avoid overly speculative computation. GVT computation is a distributed operation whose performance is limited by the communication subsystem; therefore, the communication performance effectively limits the rate of event processing while GVT is being computed. We investigate both technological and algorithmic improvements to this implementation. More specifically, we investigate the effect of using a Myrinet high-speed network with user-level communication libraries to replace the inefficient TCP/IP protocol stack and Ethernet network. In addition, we are changing the communication model for point-to-point as well as group communication operations to remove the centralized server from the critical path, significantly reducing the message latency and allowing better scalability. We are also applying system level improvements such as message aggregation and adaptive setting of message polling frequency on the receiver side.

2. Using Field-programmable Gate Array (FPGA) boards to accelerate the performance of some critical simulation subsystems. In this effort, we focus on exploiting available WildStar II boards with Xilinx Virtex 2000 FPGAs to accelerate the performance of PDES. This is a novel use of such boards—there is limited experience in using them in the context of a distributed application. Since PDES is a fine-grained operation, and the communication with the FPGA board is expensive, we needed to create an alternative channel for the FPGAs to communicate without having to interrupt the primary host processor. To achieve this, we designed a serial all-to-all connector board that provides direct, low bandwidth, low latency, connectivity among the FPGA boards. The first application we considered was the GVT computation (discussed above). We use an algorithm for the GVT computation based on Mattern's two-phase algorithm. Each node provides local time and message counts when it enters the GVT computation phase and whenever the transit message count changes to the FPGA board. The boards communicate among each other to detect the global messages in transit count. When that reaches 0, they compute the minimum of the local times and broadcast it to all the host processors. The implementation is complete and we are in the process of testing and evaluating. We are considering several other ideas using the same model. For example, the low-latency direct connectivity between the FPGAs is ideally suited for monitoring/exchange of control messages. We will look to exploit this capability to carry out adaptive control of the simulation configuration.

Both of these efforts are ongoing. Therefore, our report describes our implementations and presented some preliminary results. The remainder of this paper is organized as follows: Section 2 presents our communication subsystem effort and results; Section 3 presents the FPGA-based efforts. Finally, Section 4 presents some concluding remarks.

2. Communication Subsystem Optimization

PDES is a fine-grained application that typically uses small-sized messages to exchange events. Furthermore, it uses an optimistic synchronization model where events are processed speculatively and committed only when it is safe to do so. Additional point-to-point as well as group communication messages are necessary to achieve synchronization.

Computation time on a high performance cluster is significantly smaller than the communication time, which involves expensive buffer copies, checksum computations, and packing and unpacking of messages. Typically, event communication costs are 2–3 orders of magnitude higher than event processing costs. Furthermore, slow synchronization leads to artificial throttling of computation for time synchronization models that artificially limit optimism. Because of these reasons we decided to target the optimization of the communication subsystem.

We identified some structural as well as algorithmic improvements to the SPEEDES communication subsystem: 1) SPEEDES uses a centralized server to forward all messages and implement group-communication operations (e.g., broadcast, multicast, and reduction operations). All messages are sent first to the centralized server, and then forwarded to the destination. This effectively doubles the message latency and makes the server a bottleneck as the size of the simulation scales; 2) SPEEDES message library is built on top of TCP/IP, a heavy-weight protocol which does not run in user space and has large unnecessary overhead for operations such as checksum which are not needed in a reliable cluster environment; and 3) We identified some opportunities to dynamically amortize overheads for message sends and receives.

In our preliminary effort to address these problems, we investigated the following improvements: 1) creating direct communication channels between the simulation processes and bypassing the centralized server for point-to-point messages; and 2) investigated other approaches to dynamic message optimization such as dynamic receive event polling. Specifically, the standard implementation of SPEEDES polls the receive socket for received data with every processed event. However, when we profiled the percentage of these probes that actually returned an event, we found that the probes were successful less than 1% of the time for several applications (this varied with the process within the simulation). Since probing is expensive, we investigated reducing the period of probing such that event reception is checked once every few events processed. While this reduces the overhead, it may also harm performance if it delays the reception of a critical event; and 3) We are currently investigating using user-space message libraries and high performance communication hardware (Myrinet infrastructure and the Myrinet GM message library).

Table 1. Simulation time for different communication alternatives

Proc. #	Original Time (sec)	Direct Comm. Time (sec)	Optimized Comm. (sec)	Both (sec)
1	755.1	748.2	663.6	663.6
2	744.9	686.1	656.1	580.5
4	333.3	327.8	288.5	282.9
8	170.6	164.3	145.5	147.8
16	98.4	92.2	95	80
29	81.1	67.3	76.9	66.3

Table 1 summarizes our preliminary results for a Joint Battlespace Infrastructure publish subscribe simulation model on a Debian Linux cluster with 30 nodes each consisting of a dual Pentium 2.4GHz CPU. The column labeled "Direct Comm." shows the results with direct communication between simulation processes bypassing the centralized server (only for point-to-point messages). The column labeled "Optimized Comm." shows the results with dynamic message polling, and some other optimizations of the simulation loop. We also show the combined effects of these optimizations. Even with these preliminary implementations, significant improvement in performance is observed. We are currently refining these solutions and implementing others, including the Myrinet implementation mentioned above.

3. Utilizing FPGA Boards

The second major research effort described in this paper is the use of the FPGA boards available on the HHPC cluster at AFRL Rome. Typically, FPGA boards are used to accelerate sequential or high-granularity parallel applications that have high data parallelism or unusual data-paths (e.g., image processing or cryptographic applications). PDES does not fit this profile: it is fine-grained and does not, in general, require high data parallelism.

The FPGA boards reside on the I/O bus for the cluster and are not able to master this bus. This forces all FPGA communications to go through the primary host, making interaction with the FPGA or across FPGA boards expensive and limiting the use of the FPGAs for fine-grained applications. To remove this limitation we designed an all-to-all serial communication board that connects the FPGA boards on the different cluster nodes to each other directly through the I/O connector for the boards. This connectivity provides a direct conduit for communication between the FPGAs that does not require host interference. While this channel allows low latency communication, it is bit-serial and unsuitable for exchanging large messages.

We identified several optimizations that can take advantage of this configuration. We decided to focus first on migrating GVT computation to the FPGA boards. The GVT problem is that of computing the minimum simulation time accounting for messages in transit. SPEEDES implements a rather inefficient GVT algorithm that once a certain number of processed events is reached, continues computing the messages in transit until the message count reaches zero. At that point the minimum local simulation time becomes the GVT since there are no messages in transit. We replaced this algorithm with Mattern's algorithm[2] which uses two colors to demarcate messages sent before GVT computation is initiated from ones sent after it and avoids the need to flush messages in transit.

GVT computation significantly affects the performance of the simulator. In addition to stalling the simulation while the messages are flushed, the current estimate of GVT drives the progress rate of the simulation when optimism is being controlled. For example, the SPEEDES synchronization algorithm only allows a fixed number of events beyond the current estimate of GVT to be processed. If this number is reached, the simulation stalls awaiting GVT. Improving GVT computation overhead could thus lead to a dramatic improvement in simulation performance.

The implementation on the FPGA works as follows. Once GVT computation is initiated, a node sends updates of its Local Virtual Time (LVT), or its message in transit count to the FPGA. The FPGA stores this value and broadcasts it to the other FPGAs through the all-to-all connector. Once an update is received, the FPGA computes the total message count in transit. If the message count reaches zero, it computes the minimum of the LVT stamps and sends an interrupt to its host processor informing it of a GVT update. This implementation has been completed and is in the testing phase. We are also pursuing several other optimizations that use the FPGA.

4. Conclusions and Summary

In this paper, we report on our preliminary work in accelerating PDES on a Heterogeneous High Performance Cluster (HHPC). PDES is an important application that is of interest to many military applications. We explored improvements to the simulator that are application independent (they do not rely on the characteristics of the model) in two major directions: (1) Communication infrastructure and library improvements; and (2) Exploiting an FPGA board for PDES acceleration. We described our initial efforts in both of these avenues. We are currently pursuing several other opportunities in each direction.

References

1. Fujimoto, Richard, "Parallel Discrete Event Simulation." *Communications of ACM*, 33(10), October 1990.

2. Mattern, F., "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation." *Journal of Parallel and Distributed Computing*, Vol. 18, No. 4, 1993, p. 8.